

Binary Matroids and Quantum Probability Distributions

Dan Shepherd*

May 12, 2010

Abstract

We characterise the probability distributions that arise from quantum circuits all of whose gates commute, and show when these distributions can be classically simulated efficiently. We consider also marginal distributions and the computation of correlation coefficients, and draw connections between the simulation of stabiliser circuits and the combinatorics of representable matroids, as developed in the 1990s.

Acknowledgements

Much of this research was undertaken at the University of Bristol, in context of related work with Richard Jozsa and Michael Bremner (*cf.* [4]). The research was primarily funded by CESG.

1 Introduction

It is widely held that quantum computation is a useful paradigm for algorithmics because it enables us to describe a wider class of algorithms than can be described classically, and some of these may represent qualitatively faster solutions to certain problems than could ever be obtained classically, in principle. To put this sentiment on a more rigorous footing, one often sees the conjecture phrased in terms of classes of decision languages circumscribed by asymptotic time constraints, *e.g.* *Conjecture* : $\mathbf{BQP} \neq \mathbf{BPP}$. A more general approach looks not at *decision languages* arising from uniform families of circuits, but at *uniform families of probability distributions* arising from uniform families of circuits. Then instead of asking “Can this decision language be computed efficiently?” one asks more general questions such as “Can this probability distribution be sampled from efficiently? Can this probability distribution be *approximately* sampled from efficiently?”

*djs_at_cantab.net

Can we efficiently compute the correlation coefficients of this distribution? Given samples from the distribution, can we efficiently verify the hypothesis that they were thus sampled?” and so on. The reader is directed to [4] for more complexity-theoretic background and analysis on questions of this type.

In this paper, we restrict attention to the combinatorial analysis of a particular kind of uniform family of quantum circuits called **IQP**, first introduced in [12] and discussed at greater length in [11]. We call such circuits *X-programs* (see §3) because they are described by Hamiltonians composed entirely of Pauli *X* operators. From the perspective of implementation, these circuits can be rendered by any model for computing in the class \mathbf{QNC}_f^0 (defined in [7]). In this paper (§3), we fully describe the probability distributions that they generate, and also give formulas for the correlation coefficients associated to these distributions. We show when these probabilities and correlations are computable efficiently classically, and when they are not. We recall the argument [11, 4] for why it is generally impossible to sample efficiently classically from these distributions, unless the polynomial hierarchy collapses. This is by no means the first time a fully combinatoric treatment has been made of physical problems (*e.g.* [15]).

All of these results depend on describing X-programs in terms of binary matroids; equivalently binary linear codes. We begin by recalling the relationship between matroids and codes, and we note a link between the efficient simulation of Clifford circuits [2] and Vertigan’s algorithm for binary matroids [13]. Our aim is to provide a fairly self-contained reference, so all of the combinatorial definitions needed will be given up front in §2, where (almost) no mention will be made of quantum computation.

2 Combinatorics

This section introduces the necessary combinatorics notation and ideas. We give three propositions, together with sketch proofs. To proceed to the results of §3, the uninitiated reader will want to understand the statements of these propositions, though perhaps not necessarily learn the methods of their proofs. The key subsection here is §2.3, where the main computational complexity results are recalled. In §2.4 we give worked examples of the transformations that are used later.

Let P be any binary matrix, that is, any matrix over \mathbb{F}_2 having, say, n rows and l columns and rank r . To it we associate a *binary linear code* $\mathcal{C} = \mathcal{C}(P)$ and a *binary matroid* $\mathcal{M} = \mathcal{M}(P)$. The code \mathcal{C} will be generated by the columns of P , and the matroid \mathcal{M} will be coordinatized by the rows of P . Such codes and matroids are in one-to-one correspondence (Proposition 1 below), so everything that can be said about the one can be said about the other : accordingly we will try to say everything twice, for clarity.

2.1 Formal definitions

Formally, \mathcal{C} is defined to be the subset of the vector space \mathbb{F}_2^n that is generated by linearly combining *columns* of P . The cardinality of the code is thus 2^r , so we call r its *rank*. The elements of \mathcal{C} are called *codewords*, and these may also be thought of as binary strings of length n . The parameter n is called the *length* of the code. The parameter l is not well-defined for the code, only for its generator matrix P , though clearly $r \leq l$.

Formally, \mathcal{M} is defined to be the isomorphism class of the (multi)set E of rows of P (vectors in the vector space \mathbb{F}_2^l) together with the induced rank function $\rho_{\mathcal{M}}$ that maps subsets of these row-vectors to their rank, *i.e.* to whichever integer counts the dimension of the subspace of \mathbb{F}_2^l that they span. In particular, $\rho_{\mathcal{M}}(\emptyset) = 0$ and $\rho_{\mathcal{M}}(E) = r$, by definition. The parameter n is called the *size* of the matroid, and r is called its *rank*. The parameter l is not well-defined for the isomorphism class, since the row-vectors could easily be embedded in a larger vector space, though clearly $r \leq l$.

Proposition 1 *For binary matrices P and Q , the following are equivalent :*

- P and Q generate the same code, $\mathcal{C}(P) = \mathcal{C}(Q)$;
- P and Q generate the same matroid, $\mathcal{M}(P) = \mathcal{M}(Q)$;
- there exist binary matrices R and R' such that $P = Q \cdot R$ and $P \cdot R' = Q$.

To prove this, we give a *normal form* for a matrix P . First, identify a basis for the matroid $\mathcal{M}(P)$, that is, find r rows that collectively have rank r . Without loss of generality, *i.e.* for illustration, we assume these to be the first r rows of P . Then we can write

$$\begin{aligned} P &= \begin{pmatrix} B \\ C \end{pmatrix} = \begin{pmatrix} I \\ D \end{pmatrix} \cdot B, \\ P' &:= \begin{pmatrix} I \\ D \end{pmatrix}, \end{aligned}$$

where B is the submatrix given by the first r rows, and C is the remaining submatrix. Here I is an r -by- r identity matrix. We can find D uniquely satisfying $D \cdot B = C$ precisely because B is a basis for the space to which every row of P belongs. This kind of reduction can be achieved by *Gaussian elimination*, and P' is called an *echelon reduction* of P . Echelon reduction is unique once an ordered basis is chosen, but we are free to choose any basis with any ordering. Elsewhere in the paper we use the notation P' to denote an echelon reduction of P .

The reader may complete the proof of the Proposition, using this idea of identifying a basis set and an echelon reduction. ■

2.2 Weight enumerator and Tutte polynomial

The weight enumerator polynomial of \mathcal{C} is a monovariate polynomial (*i.e.* element of $\mathbb{Z}[\zeta]$ for indeterminate ζ) given by

$$W_{\mathcal{C}}(\zeta) := \sum_{\mathbf{c} \in \mathcal{C}} \zeta^{|\mathbf{c}|},$$

where $|\mathbf{c}|$ denotes the Hamming weight of the codeword \mathbf{c} .

The Tutte polynomial of \mathcal{M} is a bivariate polynomial (*i.e.* element of $\mathbb{Z}[x, y]$ for indeterminates x, y) given by

$$T_{\mathcal{M}}(x, y) := \sum_{X \subseteq E} (x - 1)^{\rho_{\mathcal{M}}(E) - \rho_{\mathcal{M}}(X)} \cdot (y - 1)^{|X| - \rho_{\mathcal{M}}(X)},$$

where $\rho_{\mathcal{M}} : E \rightarrow \mathbb{Z}$ is the rank function defining the matroid.

Here is a random example, with $n = 6$, $r = 3$, $l = 4$:

$$P = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix};$$

$$W_{\mathcal{C}(P)}(\zeta) = 1 + 4\zeta^2 + 3\zeta^4;$$

$$T_{\mathcal{M}(P)}(x, y) = y(x + y)(x^2 + x + y).$$

The natural connection between these two concepts is given by Greene's theorem :

Proposition 2 (*C. Greene, 1976*)

$$W_{\mathcal{C}}(\zeta) = \zeta^{n-r} \cdot (1 - \zeta)^r \cdot T_{\mathcal{M}}\left(\frac{1 + \zeta}{1 - \zeta}, \frac{1}{\zeta}\right),$$

where $n = |E|$ and $r = \rho_{\mathcal{M}}(E)$ and E is the base set for \mathcal{M} , as usual.

This can be proved inductively, using the entirely standard matroid notions of *deletion* and *contraction*, and taking for the base cases those few examples where $r \leq n \leq 1$. Note that *loops* and *coloops* are conveniently dealt with first as special cases. ■

Throughout this paper, we find it convenient to use a *normalised* version of the weight enumerator/Tutte polynomial, given in terms of a single *real* parameter θ . We fix the

following notation for use throughout the rest of the paper :

$$\begin{aligned}\alpha_{(P,\theta)} &:= 2^{-r} \cdot e^{i\theta n} \cdot W_{\mathcal{C}(P)} \left(e^{-2i\theta} \right) \\ &= e^{i\theta(r-n)} \cdot i^r \cdot \sin^r \theta \cdot T_{\mathcal{M}(P)} \left(\frac{e^{i\theta} + e^{-i\theta}}{e^{i\theta} - e^{-i\theta}}, e^{2i\theta} \right),\end{aligned}$$

again with r for the rank and n for the length (size) of the code (matroid). Note that θ is really only defined modulo 2π , and inverting the sign of θ only conjugates the output α value trivially.

Here is a useful observation, showing how normalisation ensures $|\alpha_{(P,\theta)}| \leq 1$:

Proposition 3 *The scalar $\alpha_{(P,\theta)}$ defined above can be expressed directly in terms of the code $\mathcal{C}(P)$ according to the probabilistic formula below.*

$$\alpha_{(P,\theta)} = \mathbb{E}_{\mathbf{c} \in \mathcal{C}(P)} \left[\exp(i\theta \cdot (n - 2|\mathbf{c}|)) \right].$$

To see this, observe that the factor $e^{i\theta n}$ can be pulled out in front of the expectation operator, and what is left constitutes a probabilistic interpretation of the required value $2^{-r} \cdot W_{\mathcal{C}}(e^{-2i\theta})$, because there are 2^r codewords in the code. \blacksquare

2.3 Computational Complexity

What is the computational complexity of computing the value of the scalar $\alpha_{(P,\theta)}$? This question is addressed in [8] and [13]. The multiples of $\frac{\pi}{4}$ are found all to be essentially trivial.

For example, if $\theta = \pi$ then $\alpha_{(P,\theta)} = (-1)^n$. If $\theta = \frac{\pi}{2}$ then $\alpha_{(P,\theta)}$ vanishes whenever \mathcal{C} is not an even code, and evaluates to i^n whenever it is even. One can determine whether a code is even by looking at *any* generator matrix for it : the code is not even iff any column of P has odd Hamming weight. If $\theta = \frac{\pi}{4}$ then Vertigan's algorithm [13] provides an efficient explicit polynomial-time recursion to evaluate $T_{\mathcal{M}}(-i, i)$, which is proportional to $\alpha_{(P,\frac{\pi}{4})}$. This algorithm is remarkably similar to the algorithm used in classically simulating the probability distribution associated to a Gottesman-Knill-Clifford computation [2].

Conversely, for any other values of θ , the worst-case complexity for computing $\alpha_{(P,\theta)}$ (over the class of all binary matrices P) is $\#\mathbf{P}$ -hard, and there are efficient reductions from one θ to any other (excluding multiples of $\frac{\pi}{4}$, limiting to algebraic values of $e^{i\theta}$) [8, 13]. In particular, the reader should note that $\theta = \frac{\pi}{8}$ is no less hard than any other value of θ , when it comes to evaluating $\alpha_{(P,\theta)}$ in the worst case.

The same hardness holds for specific subclasses of matroid, *e.g.* for *graphic* matroids [8]. But it should come as no surprise that there are ‘trivial’ classes of matroid where $\alpha_{(P,\theta)}$

is readily evaluated for all θ . The primary example would seem to be the case of graphic matroids having *bounded treewidth* [3]. Jaeger *et al.* also claim that $T_{\mathcal{M}}(x, y)$ is easily evaluated when $(x - 1)(y - 1) = 2$ (as is always the case for our $\alpha_{(P, \theta)}$) whenever \mathcal{M} is the cycle matroid of a *planar* graph (*cf.* [8] §(5.8), also [14] lemma 12.2, and for the main reduction, see [6]).

2.4 Transformations

There are two matroid/code transformations that we shall be requiring in §3 for our first two theorems. Notation for these is given here, but is not generally found in the literature. We include example computations of both transformations, to add clarity.

Projection

Let \mathbf{x} be a non-zero string of l bits, and from the matrix P (*resp.* the code \mathcal{C} , the matroid \mathcal{M}) we derive $P\top\mathbf{x}$ (*resp.* $\mathcal{C}\top\mathbf{x}$, $\mathcal{M}\top\mathbf{x}$) by projecting each row along direction \mathbf{x} onto a hyperplane avoiding \mathbf{x} . This projection operation is well-defined for both \mathcal{C} and \mathcal{M} , regardless of which hyperplane is chosen.

To make it well-defined for P as well, we specify (somewhat arbitrarily) that this projection is achieved by mapping a row \mathbf{a} to whichever of $\{\mathbf{a}, \mathbf{a} + \mathbf{x}\}$ is lexicographically first. Note that the rank of $\mathcal{C}\top\mathbf{x}$ (and of $\mathcal{M}\top\mathbf{x}$) will either be one less than that of \mathcal{C} (and \mathcal{M}), or else $\mathcal{C}\top\mathbf{x} = \mathcal{C}$ (and $\mathcal{M}\top\mathbf{x} = \mathcal{M}$). In either case, $\mathcal{C}\top\mathbf{x} \subseteq \mathcal{C}$ and the length is not changed. Note that if \mathbf{x} was originally a row of P , then $P\top\mathbf{x}$ contains an all-zero row, so the matroid $\mathcal{M}\top\mathbf{x}$ will contain a loop.

Here is a random example, with $n = 6$, $r = 3$, $l = 4$, and $\mathbf{x} = (0110)$:

$$P = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \rightarrow P\top\mathbf{x} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

In this example, the code/matroid is indeed changed and so the rank drops by one from 3 to 2. One more loop is created (second row) and one more non-trivial parallel class is created (first and fifth rows now parallel). Echelon-reduced forms of these example matrices

are

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow (P \top \mathbf{x})' = (P' \top \mathbf{x}')' = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

where $\mathbf{x}' = (010)$.

Affinification

Let \mathbf{s} be a string of l bits, and from the matrix P (*resp.* the code \mathcal{C} , the matroid \mathcal{M}) we derive $P_{\mathbf{s}}$ (*resp.* $\mathcal{C}_{\mathbf{s}}$, $\mathcal{M}_{\mathbf{s}}$) by deleting all rows that are orthogonal to \mathbf{s} , using the natural \mathbb{F}_2 inner-product. Note that this increases neither size (length) nor rank. Note that $\mathcal{C}_{\mathbf{s}}$ always contains the all-ones codeword, and (equivalently) $\mathcal{M}_{\mathbf{s}}$ is always an *affine* matroid, and indeed a *minor* of \mathcal{M} .

Here is a random example, with $n = 6$, $r = 3$, $l = 4$, and $\mathbf{s} = (0110)$:

$$P = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \rightarrow P_{\mathbf{s}} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

In this example, the rank did not drop, but the length (size) of the code (matroid) dropped from 6 to 4. Echelon-reduced forms of these example matrices are

$$P' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow (P_{\mathbf{s}})' = (P'_{\mathbf{s}'})' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

where $\mathbf{s}' = (101)$. Note that \mathbf{x} and \mathbf{s} transform differently, to \mathbf{x}' and \mathbf{s}' respectively, under echelon-reduction. This is because \mathbf{s} really belongs to the *dual* space, being the space of linear maps $\mathbb{F}_2^l \rightarrow \mathbb{F}_2$. (It is convenient to think of \mathbf{x} as a row-vector and \mathbf{s} as a column-vector.)

3 X-programs for IQP

The class \mathbf{QNC}_f^0 was introduced by Høyer and Spalek [7] to capture the idea of allowing only a constant amount of scope for temporal complexity within a (uniform) family of circuits. **IQP** goes ‘a step further’ by allowing essentially *no* temporal structure within the abstract quantum process. The main simulation results for **IQP** distributions (as given by X-programs) and their marginals are given in §3.5 and §3.6, respectively.

3.1 Definition

The definition given below is a little different from that given in [4], to enable the combinatorial structure to be seen more clearly.

An *X-program* (P, θ) gives a recipe for building a probability distribution. Here P denotes an n -by- l matrix over \mathbb{F}_2 and θ is some real angle. The matrix P is used to build a Hamiltonian of n commuting terms on l qubits, each term a product of Pauli X operators. (Formally we demand that $l = O(\text{poly}(n))$, though it turns out there is no point even taking $l \geq n$.)

$$\mathbf{H}_P := \sum_{a=1}^n \prod_{b=1}^l X_b^{P_{ab}}.$$

Thus the columns of P correspond to qubits, while the rows of P correspond to ‘gates’ (in a circuit) or ‘interactions’ (in a Hamiltonian) on the qubits.

The distribution is then given by

$$\mathbb{P}[\mathbf{X} = \mathbf{x}] := \left| \langle \mathbf{x} | \exp(i\theta \mathbf{H}_P) | \mathbf{0} \rangle \right|^2,$$

using the squares of the magnitudes of the transition amplitudes, as required by the Born rule for quantum processes. We call this an **IQP** probability distribution.

3.2 Implementation

We consider that a quantum circuit for sampling from this distribution contains no ‘inherent temporal structure’ because all terms in the Hamiltonian commute, and so it makes no difference (in principle) which is implemented first, or whether they are implemented simultaneously, somehow.

In [12, 11] we showed explicitly how to implement these circuits in certain theoretical architectures. For example, one such implementation uses so-called *graph states*, where P

is interpreted as (the biadjacency matrix of) a bipartite graph at whose vertices are located single qubits. (A particularly interesting *mild generalisation* of X-programs—still without introducing temporal structure—is to consider the preparation and *arbitrary* measurement of graph states, with no intervening feed-forward/adaptive control of measurement results.) But these implementation issues will not be of concern to us in the present analysis. For this paper, the Hamiltonian \mathbf{H}_P ‘takes priority’ over any circuit that might be designed to implement it, or the unitary mappings it generates.

3.3 Correlation coefficients

As well as being interested in **IQP** probability distributions and the individual probabilities $\mathbb{P}[\mathbf{X} = \mathbf{x}]$, we are also interested in the *correlation coefficients*. These are given by taking the \mathbb{F}_2 Fourier transform of the *probability vector*. In symbols, we define

$$\beta_{\mathbf{s}} := 2 \cdot \mathbb{P}[\mathbf{X} \cdot \mathbf{s} = 0] - 1,$$

and thence derive the following formulæ :

$$\begin{aligned} \mathbb{P}[\mathbf{X} \cdot \mathbf{s} = 0] &= \frac{1 + \beta_{\mathbf{s}}}{2}; \\ \mathbb{P}[\mathbf{X} = \mathbf{x}] &= 2^{-l} \cdot \sum_{\mathbf{s} \in \mathbb{F}_2^l} (-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \beta_{\mathbf{s}} \\ &= \mathbb{E}_{\mathbf{s}}[(-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \beta_{\mathbf{s}}]. \end{aligned}$$

The second line above shows that the probability distribution is entirely specified by the values $\beta_{\mathbf{s}}$, and therefore encodes $2^l - 1$ degrees of freedom (the value $\beta_{\mathbf{0}}$ is fixed as 1 for all distributions). Because a distribution encodes so much data, it can be a very cumbersome object to work with.

3.4 Relation to matroids

Here we give the theorems that show the link between probability distributions for X-programs and the weight enumerator.

Theorem 1 *For any binary matrix P and any angle θ ,*

$$\langle \mathbf{0} | \exp(i\theta \mathbf{H}_P) | \mathbf{0} \rangle = \alpha_{(P, \theta)}.$$

Moreover, for $\mathbf{x} \neq \mathbf{0}$,

$$\langle \mathbf{x} | \exp(i\theta \mathbf{H}_P) | \mathbf{0} \rangle = \alpha_{(P \top \mathbf{x}, \theta)} - \alpha_{(P, \theta)},$$

where $P \top \mathbf{x}$ denotes the projection defined in §2.4, and α is the function defined in §2.2.

This theorem has not been published before. We prove it in §4. ■

There is also a simple formula for the correlation coefficients $\beta_{\mathbf{s}}$, as follows.

Theorem 2 *Let \mathbf{s} be a string of l bits and let (P, θ) be an X -program on l qubits. Let $\beta_{\mathbf{s}}$ be the correlation coefficient associated to \mathbf{s} for the associated **IQP** probability distribution. This $\beta_{\mathbf{s}}$ is a real value, and can be specified in terms of the affinification $P_{\mathbf{s}}$, as follows.*

$$\beta_{\mathbf{s}} = \alpha_{(P_{\mathbf{s}}, 2\theta)}.$$

We prove this theorem in [12, 11], and give a slightly simpler proof in §4, for completeness. ■

3.5 Implications for simulation

In this section, we summarise a variety of observations about **IQP** probability distributions, for different values of θ and different classes of P . (Later in §3.6, we consider ‘breaking apart’ **IQP** distributions into marginal probability distributions.)

$\theta = \frac{\pi}{4}$ (completely solved)

The case $\theta = \frac{\pi}{4}$ is interesting. In that case, all values $\mathbb{P}[\mathbf{X} = \mathbf{x}]$ and $\mathbb{P}[\mathbf{X} \cdot \mathbf{s} = 0]$ can be computed efficiently using Vertigan’s algorithm, together with the theorems above. Moreover, the actual probability distribution can be efficiently sampled classically, exactly. This is because the unitary map $\exp(i\frac{\pi}{4}\mathbf{H}_P)$ can be decomposed as a product of Clifford gates, whence the Gottesman-Knill theorem applies. To see this, observe for example that

$$\exp\left(\frac{i\pi}{4} \prod_{j=1}^k X_j\right) \cdot Z_1 \cdot \exp\left(-\frac{i\pi}{4} \prod_{j=1}^k X_j\right) = -i \cdot Z_1 \cdot \prod_{j=1}^k X_j,$$

and so the Pauli group is fixed by this unitary. There is thus a link between the classical simulation of so-called *stabiliser states* [2], the fact that Clifford circuits are effectively devoid of temporal structure [5], and Vertigan’s algorithm for evaluating $T_{\mathcal{M}}(-i, i)$ efficiently [13].

The following is an analogue of the Gottesman-Knill Theorem for X -programs, giving a complete description of the probability distribution in this case :

Theorem 3 *For any X -program on l qubits, with $\theta = \frac{\pi}{4}$, there is an efficiently computed affine space S of \mathbb{F}_2^l over which the associated probability distribution is supported and*

uniform. *That is to say, for some easily-determined S ,*

$$\mathbb{P}[\mathbf{X} = \mathbf{x}] = 2^{-\dim(S)} \cdot \{\mathbf{x} \in S\}.$$

We prove this theorem in §4. ■

$\theta = \frac{\pi}{8}$ (main area of ongoing research)

Another interesting case—this time not so easy to simulate—occurs when $\theta = \frac{\pi}{8}$. A quantum circuit could efficiently sample from the distribution, of course, but now a classical computer could efficiently determine any (polynomial number of) correlation coefficients of our choosing for that distribution, because $\beta_{\mathbf{s}} = \alpha_{(P_{\mathbf{s}}, \frac{\pi}{4})}$ is efficiently computable by Vertigan’s algorithm. Therefore, a classical computer would be perfectly able to run an effective hypothesis test against a purported (sufficiently large) sample generated by the quantum circuit, *regardless of the matrix P chosen*. Such an hypothesis test could be used to distinguish this ‘null’ hypothesis from some appropriate ‘alternative’ hypothesis. This *testing paradigm* generalises our earlier work [12] and we anticipate it being useful in validating quantum computing implementations. We hope to explore the ramifications of this observation more fully in future work.

The question then arises as to whether a classical computer could sample from the distribution by some other efficient means, or else sample efficiently from some statistically close distribution. We know that $\mathbf{BPP}_{\text{path}} \neq \mathbf{PP}$ (unless of course the Polynomial Hierarchy collapses), and so via Toda’s theorem and the hardness of computing $\alpha_{(P, \frac{\pi}{8})}$ [8], we can deduce that no classical algorithm will be able to sample efficiently from the **IQP** distribution exactly, for generic P . In fact, we can use the theory of *post-selection* to derive an independent proof (independent of [8]) of the hardness of evaluating $W_{\mathcal{C}}(e^{i\pi/4})$, much as Aaronson [1] used post-selection to derive an independent proof the closure properties of **PP** (see [4] for a fuller account of this).

We leave open the possibility of an *approximate* classical sampling technique. It is important to determine just how close a classical algorithm can expect to get, with distance between distributions being measured as usual by the 1-norm (total variational distance) : this is a subject we hope to address in future research.

Arbitrary θ , but ‘special’ P

For *arbitrary* θ , Tutte polynomials can be evaluated for those matroids all of whose connected components are $O(\log n)$ in size, or indeed for *graphic* matroids with $O(\log n)$ *treewidth* [3]. This suggests that somehow there is very limited ‘data flow’ within an X-program computation.

If P is the *incidence matrix* of a graph (*i.e.* two 1s per row), then the formula of Proposition 3 may be reinterpreted

$$\alpha_{(P,\theta)} = \mathbb{E}_{\mathbf{c}} \left[\exp(i\theta \cdot (n - 2|\mathbf{c}|)) \right]$$

so that n counts the number of edges of this graph, and \mathbf{c} ranges over all possible *cuts*, with $|\mathbf{c}|$ denoting the weight of a cut (a *cut* is simply a partition of vertices into two disjoint subsets, and its *weight* is simply the number of edges that cross the partition). Vertigan [14] indicates that Kasteleyn's Theorem [9] (which gives an efficient technique for counting perfect matchings on planar graphs) can be used to establish an explicit polytime algorithm for evaluating $\alpha_{(P,\theta)}$ whenever P is the incidence matrix of a planar graph (*cf.* §2.3). The algorithm for achieving this reduction is spelled out in more detail in [6].

Equivalence modulo θ

Before moving on, we give some insight as to why the values $\frac{\pi}{4}$ and $\frac{\pi}{8}$ are particularly noteworthy...

For two different X-programs P, Q , (with the same θ value), we say that $\mathbf{H}_P \sim_{\theta} \mathbf{H}_Q$ if they generate the same unitary map, $\exp(i\theta\mathbf{H}_P) = \exp(i\theta\mathbf{H}_Q)$. This notion of equivalence can be useful if one wishes to optimise certain properties of a matrix (perhaps for implementation reasons) without wishing to alter the distribution it would generate.

Proposition 4 *If $\theta = c \cdot \pi/2^d$ for fixed integers c, d , then for any matrix P (with n rows and l columns) it is computationally efficient to find a matrix Q such that*

- $\mathbf{H}_P \sim_{\theta} \mathbf{H}_Q$;
- $\# \text{ rows of } Q = O(\text{poly}(n))$, $\# \text{ columns of } Q = l$;
- *each row of Q has at most d ones.*

To prove this, simply rewrite the Pauli operator X_j as $1 - 2x_j$, where $x_j = \frac{1-X_j}{2}$ has integer eigenvalues. Then rewrite $i\theta\mathbf{H}_P$ as a polynomial in the x_j s. In this format, the number of terms may increase by an exponential factor (since a single term, *e.g.* $X_1 \dots X_l$, may expand to form 2^l terms), but we can nonetheless efficiently list all those terms whose degree (number of x_j s) is at most d . In the expansion, the remaining terms contribute nothing to $\exp(i\theta\mathbf{H}_P)$, because their coefficient is an integer multiple of $2\pi i$, and so we may ignore them. For example

$$\begin{aligned} \frac{i\pi}{4} \cdot X_1 \cdot X_2 \cdot X_3 &= \frac{i\pi}{4} \cdot (1 - 2x_1)(1 - 2x_2)(1 - 2x_3) \\ &= \frac{i\pi}{4} \left(1 - 2x_1 - 2x_2 - 2x_3 + 4x_1x_2 + 4x_1x_3 + 4x_2x_3 \right) - 2\pi i x_1x_2x_3 \end{aligned}$$

and so if $\theta = \frac{\pi}{4}$, we can replace $X_1X_2X_3$ by $X_1X_2X_3 + (1 - X_1)(1 - X_2)(1 - X_3)$, which is of lower degree.

Therefore we take $i\theta\mathbf{H}_Q$ to be given by the polynomial $i\theta\mathbf{H}_P$ with high-degree terms (counting x_j s) removed. Rewriting this as a polynomial in X_j s, we can recover Q with the desired properties. \blacksquare

As a consequence of this, for studying $\theta = \frac{\pi}{4}$ we can make do with limiting to the class of *graphic* matroids, but for $\theta = \frac{\pi}{8}$ we should also allow matrices with three 1s per row. It is doubtful that one can infer that graphic matroids are *necessarily* trivial in this context, since hardness results are shown in [4] where the underlying matroids are all graphic (and $\theta = \frac{\pi}{8}$). Nonetheless, the number of *genuinely different* programs possible for a given n would seem to be smaller for $\theta = \frac{\pi}{4}$ or even for $\theta = \frac{\pi}{8}$ than for, say, $\frac{\pi}{3}$, because of this effective restriction on row density.

3.6 Simulating marginal distributions

What happens if we attempt to simulate only a few of the output bits from an X -program, assuming that the remaining bits from \mathbf{X} are *traced out*? For this, we need some additional concepts. Let m be a linear idempotent map on \mathbb{F}_2^l , so $m = m^2$. This is called a *projector*, and we do not assume it to be orthogonal. For a (primal) vector \mathbf{x} , if $m(\mathbf{x}) = \mathbf{x}$ then we say that \mathbf{x} is *supported by* m ; equivalently we say that \mathbf{x} is *in the range* of m . Let $|m|$ denote the dimension of the range of m , so $l - |m|$ denotes the dimension of its kernel. Let K and R denote the kernel and range of m , and let K^* and R^* denote the kernel and range of its dual (m^*), so that K^* is the perpendicular space to R , and R^* is the perpendicular space to K . (This is because if $\mathbf{x} = m(\mathbf{x})$ then $\mathbf{y} \cdot \mathbf{x} = \mathbf{y} \cdot m(\mathbf{x}) = m^*(\mathbf{y}) \cdot \mathbf{x}$, and so if also $m^*(\mathbf{y}) = \mathbf{0}$, then $\mathbf{y} \cdot \mathbf{x} = 0$, *i.e.* $\mathbf{x} \perp \mathbf{y}$, and so on.)

We say that m is *supported on* b (qu)bits if K contains $l - b$ distinct vectors of Hamming weight 1. Note that it is possible for the number of bits b on which m is supported to be larger than the rank $|m|$. Likewise, \mathbf{x} might be supported by m even if $|\mathbf{x}| > |m|$. But if the matrix representation of m were diagonal, then $b = |m|$ and we could think of m simply as *masking off* certain bit-locations that are to be traced out.

Define a *marginal* probability distribution via the random variable $m(\mathbf{X})$, according to the natural construction

$$\mathbb{P}[m(\mathbf{X}) = \mathbf{x}] := \{\mathbf{x} \in R\} \cdot \sum_{\mathbf{k} \in K} \mathbb{P}[\mathbf{X} = \mathbf{x} + \mathbf{k}].$$

Proposition 5 *Let \mathbf{X} be a random variable for the distribution of an X -program (P, θ) and let m be a projector whose range is denoted R and for which the range of m^* is denoted*

R^* . Restricted to R , the random variable $m(\mathbf{X})$ has probability distribution

$$\mathbb{P}[m(\mathbf{X}) = \mathbf{x}] = \mathbb{E}_{\mathbf{s} \in R^*} [(-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \alpha_{(P_{\mathbf{s}}, 2\theta)}].$$

This is seen using the following chain of reasoning, working from the definition above, recalling §3.3 and Theorem 2.

$$\begin{aligned} \mathbb{P}[m(\mathbf{X}) = \mathbf{x}] &= \{ \mathbf{x} \in R \} \cdot \sum_{\mathbf{k} \in K} 2^{-l} \cdot \sum_{\mathbf{s} \in \mathbb{F}_2^l} (-1)^{\mathbf{x} \cdot \mathbf{s} + \mathbf{k} \cdot \mathbf{s}} \cdot \beta_{\mathbf{s}} \\ &= 2^{-l} \cdot \{ \mathbf{x} \in R \} \cdot \sum_{\mathbf{s} \in \mathbb{F}_2^l} (-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \beta_{\mathbf{s}} \cdot \sum_{\mathbf{k} \in K} (-1)^{\mathbf{k} \cdot \mathbf{s}} \\ &= 2^{-|m|} \cdot \{ \mathbf{x} \in R \} \cdot \sum_{\mathbf{s} \in R^*} (-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \beta_{\mathbf{s}} \\ &= \{ \mathbf{x} \in R \} \cdot \mathbb{E}_{\mathbf{s} \in R^*} [(-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \alpha_{(P_{\mathbf{s}}, 2\theta)}]. \end{aligned}$$

■

Such marginal distributions are often used in defining *decision languages* from families of distributions arising from uniform families of computational circuits [4]. Marginal distributions are also useful when attempting to render classical sampling algorithms, because it is clear that if each probability of each marginal distribution is computable, then there are many ways in which samples from the distribution can be simulated efficiently.

Other combinatorial results (strong simulation of marginals)

A theorem about $\theta = \frac{\pi}{8}$ that works for arbitrary P :-

Theorem 4 *For an X -program (P, θ) with n terms and angle $\theta = \frac{\pi}{8}$, giving rise to output distribution \mathbf{X} , for any ‘masking’ projector m with $|m| = O(\log(n))$, the probability vector for the marginal distribution $m(\mathbf{X})$ is polynomially long and can be explicitly computed (classically) efficiently, regardless of P .*

A theorem for ‘sparse’ P that works for arbitrary θ :-

Theorem 5 *For an X -program (P, θ) with n terms and arbitrary angle θ , giving rise to output distribution \mathbf{X} , for any ‘masking’ projector m with $|m| = O(\log(n))$ that is actually supported on $O(\log n)$ of the qubits, the probability vector for the marginal distribution $m(\mathbf{X})$ is polynomially long and can be explicitly computed (classically) efficiently, provided there is some constant c (independent of the problem instance) bounding the number of 1s in any column of P .*

A theorem for ‘graphic’ P that works for arbitrary θ :-

Theorem 6 *For an X -program (P, θ) with n terms and arbitrary angle θ , giving rise to output distribution \mathbf{X} , for any ‘masking’ projector m with $|m| = 2$ that is actually supported on at most 2 of the qubits, the probability vector for the marginal distribution $m(\mathbf{X})$ can be explicitly computed (classically) efficiently, provided each row of P contains at most two 1s.*

Elementary proofs are given in §4. ■

Non-combinatorial results (weak simulation of marginals)

Moreover, *without considering Tutte polynomials at all*, when $|m| = 1$, and say $R = \{\mathbf{0}, \mathbf{m}\}$, $R^* = \{\mathbf{0}, \mathbf{m}^*\}$, we can always sample classically from the distribution $m(\mathbf{X})$ over 1-bit strings—*i.e.* emulate single output qubits independently—because of the formula

$$\begin{aligned} \mathbb{P}[m(\mathbf{X}) = \mathbf{0}] &= \mathbb{P}[\mathbf{X} \cdot \mathbf{m}^* = 0] \\ &= \frac{1 + \alpha_{(P_{\mathbf{m}^*}, 2\theta)}}{2} \\ &= \frac{1 + \mathbb{E}_{\mathbf{c} \in \mathcal{C}(P_{\mathbf{m}^*})} [\exp(2i\theta \cdot (n_{\mathbf{m}^*} - 2|\mathbf{c}|))]}{2} \\ &= \mathbb{E}_{\mathbf{c} \in \mathcal{C}(P_{\mathbf{m}^*})} [\cos^2(\theta \cdot (n_{\mathbf{m}^*} - 2|\mathbf{c}|))], \\ \mathbb{P}[m(\mathbf{X}) = \mathbf{m}] &= \mathbb{E}_{\mathbf{c} \in \mathcal{C}(P_{\mathbf{m}^*})} [\sin^2(\theta \cdot (n_{\mathbf{m}^*} - 2|\mathbf{c}|))], \end{aligned}$$

where $n_{\mathbf{m}^*}$ denotes the length of $\mathcal{C}(P_{\mathbf{m}^*})$ as usual. Note the use of Proposition 3 in this deduction, and the fact that $\alpha_{(P_{\mathbf{m}^*}, 2\theta)}$ is real. We can render this sample classically efficiently by sampling $\mathcal{C}(P_{\mathbf{m}^*})$ uniformly, even though we need have no idea how to *compute* the value $\alpha_{(P_{\mathbf{m}^*}, 2\theta)}$.

This result can be generalised to the following theorem :

Theorem 7 *For an X -program (n terms by l qubits) with any θ , if m is a projector on \mathbb{F}_2^l inducing a marginal probability distribution, with $|m| = O(\log(n))$ there is then an efficient (*i.e.* $O(\text{poly}(n))$ time) classical algorithm for sampling from this marginal distribution, accurate to exponential precision.*

This is proved in [4] without direct reference to combinatorics. We also provide a proof below in §4 without direct reference to quantum computing. ■

4 Appendix of Proofs

The proofs of Theorems 1, 2, 3, 4, 5, 6, 7 are given in this section.

Proof of Theorem 1

This Theorem directly links probabilities for X-programs to Tutte polynomials.

First we make a change of basis, writing

$$\langle \mathbf{0} | \exp(i\theta \mathbf{H}_P) | \mathbf{0} \rangle = \langle + | \exp \left(i\theta \sum_{a=1}^n \prod_{b=1}^l Z_b^{P_{ab}} \right) | + \rangle.$$

Because this contains a matrix that is patently diagonal, and because its vectors $\langle + |$ and $| + \rangle$ represent uniform superpositions (with matching phase), we can assert that this expression evaluates to

$$\begin{aligned} 2^{-l} \cdot \text{Tr} \left[\exp \left(i\theta \sum_{a=1}^n \prod_{b=1}^l Z_b^{P_{ab}} \right) \right] &= 2^{-l} \cdot \sum_{\mathbf{s} \in \mathbb{F}_2^l} \exp \left(i\theta \sum_{a=1}^n \prod_{b=1}^l (-1)^{P_{ab} \cdot s_b} \right) \\ &= \mathbb{E}_{\mathbf{s} \in \mathbb{F}_2^l} \left[\exp \left(i\theta \sum_{a=1}^n (-1)^{(P \cdot \mathbf{s})_a} \right) \right] \\ &= \mathbb{E}_{\mathbf{c} \in \mathcal{C}(P)} \left[\exp \left(i\theta \sum_{a=1}^n (-1)^{c_a} \right) \right] \\ &= 2^{-r} \cdot \sum_{\mathbf{c} \in \mathcal{C}(P)} \exp(i\theta(n - 2|\mathbf{c}|)) \\ &= \alpha_{(P, \theta)}. \end{aligned}$$

To evaluate the other transition amplitudes, we need to observe that

$$\langle \mathbf{x} | = \langle \mathbf{0} | \exp \left(i\frac{\pi}{2} \left(-1 + \prod_{b=1}^l X_b^{x_b} \right) \right).$$

The case $\theta = \frac{\pi}{2t}$ is most easily dealt with, where $t \in \mathbb{Z}^+$. Then let $Q(j)$ denote a matrix obtained from P by appending j extra rows identical to \mathbf{x} , and we quickly see that

$$\begin{aligned} \langle \mathbf{x} | \exp \left(i\theta \sum_{a=1}^n \prod_{b=1}^l X_b^{P_{ab}} \right) | \mathbf{0} \rangle &= e^{-i\theta t} \cdot \langle \mathbf{0} | \exp \left(i\theta \sum_{a=1}^{n+t} \prod_{b=1}^l X_b^{Q(t)_{ab}} \right) | \mathbf{0} \rangle \\ &= -i \cdot \alpha_{(Q(t), \theta)}. \end{aligned}$$

Now using the fact that $\alpha_{(Q(j), \theta)}$ is proportional to the evaluation of some Tutte polynomial, we can use the deletion and contraction formulæ to remove the extra rows that were added, according to the following chain of reasoning. Let $R(j)$ denote the matrix $P \top \mathbf{x}$ with j extra

loops (all-zero rows) appended. This corresponds to the matroid obtained by contracting one of the appended rows from $Q(j+1)$.

There are two sub-cases to consider. In the first, the size and rank of $\mathcal{M}(Q(t))$ are $n+t$ and r respectively, and \mathbf{x} lies within the span of E , and $\mathcal{C}(P \top \mathbf{x})$ is strictly contained within $\mathcal{C}(P)$. We deduce

$$\begin{aligned} T_{\mathcal{M}(Q(j))}(x, y) &= T_{\mathcal{M}(Q(j-1))}(x, y) + T_{\mathcal{M}(R(j-1))}(x, y), \\ T_{\mathcal{M}(R(j))}(x, y) &= y^j \cdot T_{\mathcal{M}(P \top \mathbf{x})}(x, y), \end{aligned}$$

and hence

$$T_{\mathcal{M}(Q(t))}(x, y) = \frac{1-y^t}{1-y} \cdot T_{\mathcal{M}(P \top \mathbf{x})}(x, y) + T_{\mathcal{M}(P)}(x, y).$$

Substituting this into our formula for α , and setting $x = \frac{e^{i\theta} + e^{-i\theta}}{e^{i\theta} - e^{-i\theta}}$ and $y = e^{2i\theta}$, we obtain

$$\begin{aligned} -i \cdot \alpha_{(Q(t), \theta)} &= -i \cdot e^{i\theta(r-n-t)} \cdot i^r \cdot \sin^r \theta \cdot \left(\frac{1-e^{2i\theta t}}{1-e^{2i\theta}} \cdot T_{\mathcal{M}(P \top \mathbf{x})}(x, y) + T_{\mathcal{M}(P)}(x, y) \right) \\ &= -e^{i\theta(r-n)} \cdot i^r \cdot \sin^r \theta \cdot \left(\frac{2}{1-e^{2i\theta}} \cdot T_{\mathcal{M}(P \top \mathbf{x})}(x, y) + T_{\mathcal{M}(P)}(x, y) \right) \\ &= e^{i\theta(r-1-n)} \cdot i^{r-1} \cdot \sin^{r-1} \theta \cdot T_{\mathcal{M}(P \top \mathbf{x})}(x, y) \\ &\quad - e^{i\theta(r-n)} \cdot i^r \cdot \sin^r \theta \cdot T_{\mathcal{M}(P)}(x, y) \\ &= \alpha_{(P \top \mathbf{x}, \theta)} - \alpha_{(P, \theta)}. \end{aligned}$$

The second sub-case has that the size and rank of $\mathcal{M}(Q(t))$ are $n+t$ and $r+1$ respectively, and \mathbf{x} does not lie within the span of E , and $\mathcal{C}(P \top \mathbf{x}) = \mathcal{C}(P)$. We deduce

$$\begin{aligned} T_{\mathcal{M}(Q(1))}(x, y) &= x \cdot T_{\mathcal{M}(P)}(x, y), \\ T_{\mathcal{M}(Q(j))}(x, y) &= T_{\mathcal{M}(Q(j-1))}(x, y) + T_{\mathcal{M}(R(j-1))}(x, y), \\ T_{\mathcal{M}(R(j))}(x, y) &= y^j \cdot T_{\mathcal{M}(P \top \mathbf{x})}(x, y), \end{aligned}$$

and hence

$$T_{\mathcal{M}(Q(t))}(x, y) = \left(\frac{1-y^t}{1-y} + x - 1 \right) \cdot T_{\mathcal{M}(P)}(x, y).$$

On substituting in $x = \frac{e^{i\theta} + e^{-i\theta}}{e^{i\theta} - e^{-i\theta}}$ and $y = e^{2i\theta}$, we see that the expression vanishes. It therefore still holds (now somewhat vacuously) that

$$-i \cdot \alpha_{(Q(t), \theta)} = \alpha_{(P \top \mathbf{x}, \theta)} - \alpha_{(P, \theta)}.$$

With both sub-cases established, we argue that since this equation holds whenever $\theta = \frac{\pi}{2t}$ for $t \in \mathbb{Z}^+$, so it must hold for all θ . This is true because the expressions can be thought of as Laurent polynomials in $z = e^{2i\theta} - 1$ (*i.e.* elements of $\mathbb{C}[z, z^{-1}]$, because $y = z + 1$ and $x = 1 + 2z^{-1}$), and if two Laurent polynomials agree at an infinite number of distinct places, then they must agree everywhere. \blacksquare

Proof of Theorem 2

The proof of this Theorem first appears in [12], though the present version is a little simpler.

We start with the basic definition of $\mathbb{P}[\mathbf{X} = \mathbf{x}]$, and change into the diagonal basis in order to make the summation and lose the *bra-ket* notation. Then we work on removing the modulus signs. This gives

$$\begin{aligned}
\mathbb{P}(\mathbf{X} = \mathbf{x}) &= \left| \langle \mathbf{x} | \exp \left(i\theta \sum_{a=1}^n \prod_{b=1}^l X_b^{P_{ab}} \right) | \mathbf{0} \rangle \right|^2 \\
&= \left| 2^{-l} \sum_{\mathbf{a} \in \mathbb{F}_2^l} (-1)^{\mathbf{x} \cdot \mathbf{a}} \langle \mathbf{a} | \exp \left(i\theta \sum_{a=1}^n \prod_{b=1}^l Z_b^{P_{ab}} \right) \sum_{\mathbf{b} \in \mathbb{F}_2^l} | \mathbf{b} \rangle \right|^2 \\
&= \left| 2^{-l} \sum_{\mathbf{a} \in \mathbb{F}_2^l} (-1)^{\mathbf{x} \cdot \mathbf{a}} \exp \left(i\theta \sum_{a=1}^n (-1)^{P_a \cdot \mathbf{a}} \right) \right|^2 \\
&= \left| \mathbb{E}_{\mathbf{a}} \left[(-1)^{\mathbf{x} \cdot \mathbf{a}} \exp \left(i\theta \sum_{a=1}^n (-1)^{P_a \cdot \mathbf{a}} \right) \right] \right|^2 \\
&= \mathbb{E}_{\mathbf{a}, \mathbf{d}} \left[(-1)^{\mathbf{x} \cdot \mathbf{d}} \exp \left(i\theta \sum_{a=1}^n (-1)^{P_a \cdot \mathbf{a}} \left(1 - (-1)^{P_a \cdot \mathbf{d}} \right) \right) \right].
\end{aligned}$$

Next, we take the definition for $\beta_{\mathbf{s}}$ in terms of the Fourier transform, drop in the formula

above, and then just kill off the spare variables.

$$\begin{aligned}
\beta_{\mathbf{s}} &= \sum_{\mathbf{x} \in \mathbb{F}_2^l} (-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \mathbb{P}(\mathbf{X} = \mathbf{x}) \\
&= \sum_{\mathbf{x} \in \mathbb{F}_2^l} (-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \mathbb{E}_{\mathbf{a}, \mathbf{d}} \left[(-1)^{\mathbf{x} \cdot \mathbf{d}} \exp \left(i\theta \sum_{a=1}^n (-1)^{P_a \cdot \mathbf{a}} (1 - (-1)^{P_a \cdot \mathbf{d}}) \right) \right] \\
&= \mathbb{E}_{\mathbf{a}, \mathbf{d}} \left[2^l \cdot \{\mathbf{s} = \mathbf{d}\} \cdot \exp \left(i\theta \sum_{a=1}^n (-1)^{P_a \cdot \mathbf{a}} (1 - (-1)^{P_a \cdot \mathbf{d}}) \right) \right] \\
&= \mathbb{E}_{\mathbf{a}} \left[\exp \left(i\theta \sum_{a=1}^n (-1)^{P_a \cdot \mathbf{a}} (1 - (-1)^{P_a \cdot \mathbf{s}}) \right) \right].
\end{aligned}$$

Now it is clear that whenever $P_a \cdot \mathbf{s} = 0$, then the corresponding term vanishes, independent of the variable \mathbf{a} . Thus we need only include in the sum those terms for which $P_a \cdot \mathbf{s} = 1$.

$$\beta_{\mathbf{s}} = \mathbb{E}_{\mathbf{a} \in \mathbb{F}_2^l} \left[\exp \left(2i\theta \sum_{P_a \cdot \mathbf{s} = 1} (-1)^{P_a \cdot \mathbf{a}} \right) \right].$$

This formula we can naturally express in terms of the code $\mathcal{C}(P_{\mathbf{s}})$. Let $n_{\mathbf{s}}$ be the size of the affinified matroid (equivalently, the length of the resulting code). Use Proposition 3 to finish the deduction.

$$\begin{aligned}
\beta_{\mathbf{s}} &= \mathbb{E}_{\mathbf{c} \in \mathcal{C}(P_{\mathbf{s}})} [\exp (2i\theta(n_{\mathbf{s}} - 2|\mathbf{c}|))] \\
&= \alpha_{(P_{\mathbf{s}}, 2\theta)}.
\end{aligned}$$

■

Proof of Theorem 3

This Theorem and its proof are essentially a reworking of Vertigan's algorithm for $T_{\mathcal{M}}(-i, i)$ on binary matroids (because of Theorem 1), but here expressed more directly in terms of X-programs and their distributions. (Note that Vertigan's algorithm [13] effectively computes $\alpha_{(P, \frac{\pi}{4})}$ for input P , whereas the algorithm presented below discovers probabilities, and hence only $|\alpha_{(P, \frac{\pi}{4})}|$.)

Let \mathbf{x} be any element of \mathbb{F}_2^l and consider

$$\begin{aligned}
\mathbb{P}[\mathbf{X} = \mathbf{x}] &= \mathbb{E}_{\mathbf{s} \in \mathbb{F}_2^l} [(-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \alpha_{(P_{\mathbf{s}}, \frac{\pi}{2})}] \\
&= \mathbb{E}_{\mathbf{s} \in \mathbb{F}_2^l} [(-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot 2^{-r_{\mathbf{s}}} \cdot i^{n_{\mathbf{s}}} \cdot W_{\mathcal{C}(P_{\mathbf{s}})}(-1)].
\end{aligned}$$

The expression $2^{-r_s} \cdot W_{\mathcal{C}(P_s)}(-1)$ has some particularly simple interpretations. Evidently, if $\mathcal{C}(P_s)$ is an even code, then this expression takes the value 1; otherwise it vanishes. But the following conditions are all equivalent :

- $\mathcal{C}(P_s)$ is an even code;
- the column vector $P \cdot s$ is orthogonal to every codeword of $\mathcal{C}(P)$;
- $(P \cdot s)^T \cdot P$ is all-zero;
- $s \in \text{Ker}(P^T \cdot P)$.

Accordingly, we can restrict attention to those $s \in \text{Ker}(P^T \cdot P)$. This is simply a linear constraint on s , because the kernel of a linear map is a subspace of its domain. Write V as short-hand for $\text{Ker}(P^T \cdot P)$. Observing that n_s is given by $|P \cdot s|$ (the Hamming weight of the column vector $P \cdot s$), we see that it is indeed always even in the case $s \in V$, as one might expect.

Then define the subspace $U := \{ s : s \in V, i^{|P \cdot s|} = 1 \} \leq V$, consisting of those elements s of V for which 4 divides n_s . (Note that U really is a subspace, because it is closed under addition of elements, because for all $s \in V$, $P \cdot s$ is orthogonal to everything of the form $P \cdot a$.)

Combining these ideas,

$$\begin{aligned} \mathbb{P}[\mathbf{X} = \mathbf{x}] &= 2^{-l} \cdot \sum_{s \in \mathbb{F}_2^l} (-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot 2^{-r_s} \cdot i^{n_s} \cdot W_{\mathcal{C}(P_s)}(-1) \\ &= 2^{-l} \cdot \sum_{s \in V} (-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot i^{n_s} \\ &= 2^{-l} \cdot \left(2 \sum_{s \in U} (-1)^{\mathbf{x} \cdot \mathbf{s}} - \sum_{s \in V} (-1)^{\mathbf{x} \cdot \mathbf{s}} \right). \end{aligned}$$

Now this can be simplified further, because each of these latter sums either vanishes or else is equal to the number of terms it contains. Finally, we consider this expression in two different cases.

First case :- If $U = V$ (because 4 divides n_s for every $s \in V$) then the expression simplifies to

$$\begin{aligned} \mathbb{P}[\mathbf{X} = \mathbf{x}] &= 2^{-l} \cdot \left(\sum_{s \in V} (-1)^{\mathbf{x} \cdot \mathbf{s}} \right) \\ &= 2^{\dim(V)-l} \cdot \{ \mathbf{x} \perp V \}, \end{aligned}$$

which means that the distribution is uniform over the subspace of vectors orthogonal to the kernel of $P^T \cdot P$. We could naturally denote this ‘support’ set $S_1 := V^\perp$. Because $\mathbf{0} \in V^\perp$, this case leads to $|\alpha_{(P, \frac{\pi}{4})}| = 2^{\frac{\dim(V)-l}{2}}$.

Second case :- If $U < V$ (because only half of the elements of V have that 4 divides $n_{\mathbf{s}}$) then the expression simplifies to

$$\begin{aligned} \mathbb{P}[\mathbf{X} = \mathbf{x}] &= 2^{\dim(V)-l} \cdot (\{ \mathbf{x} \perp U \} - \{ \mathbf{x} \perp V \}) \\ &= 2^{\dim(V)-l} \cdot \{ \mathbf{x} \perp U \} \cdot \{ \mathbf{x} \not\perp V \}, \end{aligned}$$

which means that the distribution is uniform over the affine space comprising the (unique) non-trivial coset of V^\perp in U^\perp , equivalently denoted $S_2 := U^\perp \setminus V^\perp$. Because $\mathbf{0} \notin U^\perp \setminus V^\perp$, this case leads to $\alpha_{(P, \frac{\pi}{4})} = 0$. Considering our example P from §2.2 : $l = 4$, $V = \langle (1001), (0111) \rangle$, $U = \langle (0111) \rangle$, and indeed $W_{\mathcal{C}(P)}(-i) = 0$.

The two cases (S_1, S_2) are distinguished by computing (bases for) V and U . It is clear that V is readily computed. To find U (and thus establish the proof) one need only examine a basis for V . For a fixed basis, if every basis element \mathbf{s} of V has that 4 divides $n_{\mathbf{s}}$, then by closure it must be that $U = V$. Alternatively, if some ‘bad’ subset of the basis has elements \mathbf{s} for which 4 does not divide $n_{\mathbf{s}}$, then by a simple elimination technique we can change the basis (adding one ‘bad’ element into another to make it ‘good’) so that exactly one basis element has 4 not dividing $n_{\mathbf{s}}$. With this one ‘bad’ element removed, the remaining set forms a basis for U . ■

Proof of Theorem 4

Certainly when $|m| = O(\log n)$, the number of degrees of freedom associated to the distribution for $m(\mathbf{X})$ scales only polynomially in n .

If also $\theta = \frac{\pi}{8}$ then the entirety of this distribution becomes classically computable and simulable. This is achieved by running Vertigan’s algorithm a polynomial number of times to compute the $\beta_{\mathbf{s}}$ values explicitly for all $\mathbf{s} \in R^*$, in accordance with Proposition 5 : then we compute the Fourier transform of that vector to find the probability vector for the distribution in question. ■

Proof of Theorems 5 and 6

As above, $|m| = O(\log n)$ implies that we shall need to compute only polynomially many correlation coefficients ($\beta_{\mathbf{s}}$). Moreover, the condition on m actually being supported on $O(\log n)$ of the qubits (for Theorem 5) means that we need compute $\beta_{\mathbf{s}}$ only for cases

where $|\mathbf{s}| = O(\log n)$. Recall (Theorem 2 & Proposition 2) that $\beta_{\mathbf{s}}$ depends on the Tutte polynomial, rank, and size of $\mathcal{M}(P_{\mathbf{s}})$.

For any $\mathbf{s} \in \mathbb{F}_2^l$ with $|\mathbf{s}| = O(\log n)$, the size (and hence rank) of $\mathcal{M}(P_{\mathbf{s}})$ evidently cannot be bigger than $c \cdot O(\log n)$, where c bounds the number of 1s in every column of P (this can be seen *e.g.* by bounding the number of 1s there can be within the columns indicated by \mathbf{s} , and each row of $P_{\mathbf{s}}$ must contain at least one of these 1s). Even a ‘dumb’ (exponentially slow) algorithm for evaluating Tutte polynomials will require only time scaling polynomially in n^c for such tiny matroids. Hence all the required correlation coefficients, and thence the entire marginal probability distribution, can be computed as required for Theorem 5.

Note that neither the bounds on the Hamming weight of rows/columns of P nor the conditions regarding m being supported on some number of qubits are matroid invariants : rather, they depend critically on the actual presentation of P and m and are not preserved under basis transformation.

For Theorem 6, we have at most three correlation coefficients to compute (since the range of m has dimension at most two, and hence cardinality at most 4, but $\beta_{\mathbf{0}}$ is always 1). In each case, $\mathcal{M}(P_{\mathbf{s}})$ is the cycle matroid of some bipartite graph one of whose partitions contains at most two vertices (corresponding to the qubits selected by m). It is relatively straightforward to come up with a polytime algorithm for evaluating the Tutte polynomial of such an easy graph.

We will not bore the reader with an explicit derivation of such an algorithm. Suffice it to say that it is relatively straightforward to identify and group the graph minors intelligently, so as to make recursive evaluation of the Tutte polynomial efficient. This is because every minor of such a graph—having once had all its coloops deleted—is never more than two edge-contractions away from a ‘star-shaped’ graph, whose Tutte polynomial takes the explicit form

$$T_{\text{star}(\mathbf{a})}(x, y) = \prod_{j=1}^k \left(\frac{y^{a_j} - 1}{y - 1} + x - 1 \right),$$

for some integer tuple \mathbf{a} describing the star-graph. It is likely that this sort of construction could be generalised (presumably in terms of *treewidth*), extending the Theorem to cope with $|m| > 2$ (at the cost of much ink), but no such analysis has yet been made. ■

Proof of Theorem 7

We already saw (*cf.* Proposition 5, Theorem 2, and Proposition 3) that for any projector m on \mathbb{F}_2^l (range denoted R), for \mathbf{x} restricted to R ,

$$\begin{aligned}\mathbb{P}[m(\mathbf{X}) = \mathbf{x}] &= \mathbb{E}_{\mathbf{s} \in R^*} [(-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \beta_{\mathbf{s}}] \\ &= \mathbb{E}_{\mathbf{s} \in R^*} [(-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \mathbb{E}_{\mathbf{c} \in \mathcal{C}(P_{\mathbf{s}})} [\cos(2\theta \cdot (n_{\mathbf{s}} - 2|\mathbf{c}|))]] ,\end{aligned}$$

where $n_{\mathbf{s}}$ denotes the length of $\mathcal{C}(P_{\mathbf{s}})$ as usual. Conceptually breaking the code $\mathcal{C}(P_{\mathbf{s}})$ into the direct sum of two pieces across the boundary created by the mask m , we write the codeword $\mathbf{c} \in \mathcal{C}(P_{\mathbf{s}})$ as the sum of $P_{\mathbf{s}} \cdot \mathbf{t}$ and $P_{\mathbf{s}} \cdot \mathbf{k}$, for \mathbf{t} and \mathbf{k} in R^* and K^* , respectively. Hence we find that another way to write the same distribution is

$$\mathbb{P}[m(\mathbf{X}) = \mathbf{x}] = \mathbb{E}_{\mathbf{k} \in K^*} \mathbb{E}_{\mathbf{s}, \mathbf{t} \in R^*} \left[(-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \cos\left(2\theta \cdot (n_{\mathbf{s}} - 2|P_{\mathbf{s}} \cdot (\mathbf{t} + \mathbf{k})|)\right) \right] .$$

Therefore, for any $\mathbf{k} \in K^*$, let us define

$$\mathbb{P}[m(\mathbf{X}) = \mathbf{x} \mid \mathbf{k}] := \mathbb{E}_{\mathbf{s}, \mathbf{t} \in R^*} \left[(-1)^{\mathbf{x} \cdot \mathbf{s}} \cdot \cos\left(2\theta \cdot (n_{\mathbf{s}} - 2|P_{\mathbf{s}} \cdot (\mathbf{t} + \mathbf{k})|)\right) \right] .$$

Observe that this expression may be rewritten as follows.

$$\mathbb{P}[m(\mathbf{X}) = \mathbf{x} \mid \mathbf{k}] = \left| \mathbb{E}_{\mathbf{t} \in R^*} \left[(-1)^{\mathbf{x} \cdot \mathbf{t}} \cdot \exp \left(i\theta \cdot \sum_{j=1}^n (-1)^{P_j \cdot (\mathbf{t} + \mathbf{k})} \right) \right] \right|^2 .$$

(The computation proceeds along similar lines to the one in the proof of Theorem 2, but in reverse.) And from here we see that this genuinely describes a probability distribution, *i.e.* is non-negative for all \mathbf{x} .

It then follows immediately that the distribution for $m(\mathbf{X})$ is just the uniform combination of the distributions above, over $\mathbf{k} \in K^*$. Moreover, when $|m| = O(\log(n))$, the dimension of the range R^* is sufficiently small that there are only polynomially many \mathbf{t} to worry about. To sample from $m(\mathbf{X})$ one need therefore only choose \mathbf{k} uniformly from K^* and then explicitly compute the entire probability vector shown above for that value of \mathbf{k} , to an appropriate level of accuracy. Provided we use a fresh vector \mathbf{k} for each sample, the simulation will be exact, up to accuracy of the numerical precision of the real computation. ■

References

- [1] Aaronson, S. (2005). Quantum computing, postselection, and probabilistic polynomial-time, *Proc. Royal Soc. A*, 461 pp 3473–3483. ([quant-ph/0412187](#))

- [2] Aaronson, S. & Gottesman, D. (2004). Improved Simulation of Stabilizer Circuits, ([quant-ph/0406196](#))
- [3] Andrzejak, A. (1995). A polynomial-time algorithm for computation of the Tutte polynomials of graphs of bounded treewidth, *Serie B Informatik*.
- [4] Bremner, M. & Jozsa, R. & DJS. (2010). Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. ([arXiv/1005.1407](#))
- [5] Browne, D. E. & Briegel, H. J. (2006). Oneway Quantum Computation, *Lectures on quantum information*, pp 359–380. ([quant-ph/0603226](#))
- [6] Fisher, M. E. (1966). On the Dimer Solution of Planar Ising Models, *J. Math. Phys.* Vol. 7, No. 10, pp 1776–1781.
- [7] Høyer, P. & Spalek, R. (2005). Quantum circuits with unbounded fan-out, *Theory of Computing*, 1, no. 5, pp 81–103. ([quant-ph/0208043](#))
- [8] Jaeger, F. & Vertigan, D. L. & Welsh, D. J. A. (1990). On the computational complexity of the Jones and Tutte polynomials, *Math. Proc. Camb. Phil. Soc.* **108**, 35.
- [9] Kasteleyn, P. W. (1967). Graph theory and crystal physics, in Harary, F., *Graph Theory and Theoretical Physics*, New York: Academic Press, pp 43–110.
- [10] Oxley, James. *Matroid Theory*. Oxford University Press (2006).
- [11] DJS. (2009). *Quantum Complexity : restrictions on algorithms and architectures*, (Ch 5) PhD thesis, University of Bristol.
- [12] DJS. & Bremner, M. J. (2008). Temporally unstructured quantum computation, *Proc. Royal Soc. A* **465**, pp 1413–1439, doi: 10.1098/rspa.2008.0443. ([arXiv.org/0809.0847](#))
- [13] Vertigan, D. (1998). Bicycle Dimension and Special Points of the Tutte Polynomial, *Journal of Combinatorial Theory*, **B 74**, pp 378–396.
- [14] Vertigan, D. (2006/1989). The Computational Complexity of Tutte Invariants for Planar Graphs, *Siam J. Comput.*, Vol. 35, No. 3, pp 690–712.
- [15] Welsh, D. J. A., (1990). The Computational Complexity of Some Classical Problems from Statistical Physics, *Disorder in physical systems* pp 307–321, - Oxford University Press, USA.